PREDICCIÓN NÚMERO DE VEHÍCULOS INVOLUCRADOS EN ACCIDENTES PARA EL REINO UNIDO

POR:

Juan Diego Arroyave Murillo

Sebastián Sánchez Álvarez

MATERIA:

Introducción a la Inteligencia Artificial

PROFESOR:

Raúl Ramos Pollan

UNIVERSIDAD DE ANTIOQUIA

FACULTAD DE INGENIERÍA

MEDELLÍN 2023

Contenido

INTRODUCCIÓN

La inteligencia artificial (IA) es una tecnología que está cada vez más presente en nuestras vidas. Se utiliza en una amplia gama de aplicaciones, desde la medicina hasta la industria, pasando por el entretenimiento. La IA es una herramienta que puede generar mucho valor a partir de los datos. Estos datos pueden provenir de cualquier fuente, como sensores, registros históricos o encuestas. Al entrenar algoritmos de aprendizaje automático (ML) con estos datos, podemos generar modelos que nos ayudan a comprender el mundo que nos rodea.

Los modelos de ML pueden utilizarse para observar comportamientos, reducir el tamaño de la información sin perder su significancia, o generar predicciones. Las predicciones son especialmente útiles para la toma de decisiones en el corto, mediano o largo plazo.

1. Planteamiento del problema

Los accidentes de tráfico son un problema importante en el Reino Unido, con un promedio de 5,5 millones de accidentes cada año. El costo social de los accidentes de tráfico es significativo, alcanzando los 23.000 millones de libras esterlinas al año. El objetivo de este estudio es desarrollar un modelo predictivo que estime el número de vehículos involucrado en accidentes de tráfico en el Reino Unido en 2023. El modelo se basará en datos históricos enlistados en un dataset de accidentalidad entre 2005 y 2012 para Reino Unido, hallado en la plataforma de Kaggle.

En este trabajo, exploraremos la implementación de algoritmos de ML en la predicción del número de vehículos involucrados en accidentes de tráfico en el Reino Unido. Este es un tema de interés ya que la predicción de accidentes, basada en diferentes factores como los límites de velocidad, puede ser de gran utilidad para analizar las tasas de accidentalidad y generar campañas de prevención.

Para realizar esta investigación, utilizaremos un conjunto de datos que contiene información sobre accidentes de tráfico en el Reino Unido durante un período de cinco años. Este conjunto de datos incluye información sobre los factores que pueden influir en la probabilidad de un accidente, como la hora del día, el tipo de carretera y las condiciones meteorológicas. Entrenaremos diferentes algoritmos de ML con este conjunto de datos para evaluar su capacidad para predecir el número de vehículos involucrados en accidentes. Los resultados de esta investigación nos permitirán determinar si la IA puede ser una herramienta eficaz para la prevención de accidentes de tráfico.

1.1. Dataset

El marco de datos que se va a utilizar para el análisis de la problemática se extrajo de una competencia de kaggle en la cual se proporcionan datos de ocho años de información de diferentes variables relacionadas a más de treinta mil accidentes registrados en el país de Reino Unido. El dataset, se encuentra en un archivo en formato .csv que proporciona la información requerida.

El archivo que contiene los datos de los accidentes es nombrado como *road-accident-united-kingdom-uk-dataset* extraído del url: dataset y contiene la siguiente información:

- ***Accident_Index*** – Identificador para cada uno de los accidentes
- ***Location_Easting_OSGR*** – Ubicación al este

- ***Location_Northing_OSGR*** – Ubicación al norte
- ***Longitude*** – Longitud relacionada al accidente
- ***Latitude*** – Latitud relacionada al accidente
- ***Police_Force*** – Fuerza policial
- ***Accident_Severity*** – Severidad del accidente
- ***Number_of_Vehicles*** – Número de vehículos involucrados en el accidente
- ***Number_of_Casualties*** – Número de victimas
- ***Date*** – Fecha
- ***Day_of_Week*** – Día de la semana
- ***Time*** – Hora
- ***Local_Authority_(District)*** – Autoridad local del distrito
- ***1st_Road_Class*** - 1ª clase de carretera
- ***1st_Road_Number -*** 1ª clase de número
- ***Road_Type*** – Tipo de carretera
- ***Speed_limit*** – Límite de velocidad en la carretera
- ***Junction_Control*** - Control de cruces
- ***2nd_Road_Class*** - 2ª clase de carretera
- ***2nd_Road_Number*** - 2ª clase de número
- ***Pedestrian_Crossing-Human_Control*** - Control humano de los pasos de peatones
- ***Pedestrian_Crossing-Physical_Facilities*** – Instalaciones físicas para el paso de peatones
- ***Light_Conditions*** – Condiciones de la luz
- ***Weather_Conditions*** – Condiciones climáticas
- ***Road_Surface_Conditions*** – Condiciones de la carretera
- ***Special_Conditions_at_Site*** – Condiciones especiales en el sitio
- ***Carriageway_Hazards*** – Peligros en la calzada
- ***Urban_or_Rural_Area*** – Área rural o urbana
- ***Did_Police_Officer_Attend_Scene_of_Accident*** – Si el oficial de la policía atendió la escena del accidente
- ***LSOA_of_Accident_Location*** – Código de georreferenciación de la locación del accidente
- ***Year*** - Año

1.2. Métrica

La métrica de evaluación principal para el modelo será el Error Porcentual Absoluto Medio (MAPE) con el fin de minimizar el error porcentual medio de las predicciones. Esta métrica se elige dado que las variables para el modelo son cuantitativas.

1.3. Variable Objetivo

La variable objetivo que se desea predecir es “Number_of_Vehicles”, la cual nos brinda información de la cantidad de vehículos que se encuentran relacionados en cada uno de los accidentes, se analizará y posteriormente, se decidirá cuales variables serán las entradas para el entrenamiento de los algoritmos.

2. Tratamiento de datos

- **Eliminación de columnas innecesarias:**
  Dentro de la exploración del dataset, se descubrió que este incluía una columna sin título en la primera posición, la cual le proporciona un índice a cada uno de los registros que se encontraban en la base de datos. Al momento de importar el dataset a Phyton en el entorno de Colab, se decidió eliminar dicha columna sin nombre, dado que en la importación se le agregaba este índice de manera automática, por lo que resultaba innecesario tener esto de manera duplicada. En la *Figura 1* se enseña el código utilizado para eliminar dicha columna.

```
[ ]  1 #Se elimina columna con los indices, ya que pandas los agrega automaticamente
     2 df = df.drop('Unnamed: 0', axis = 1)
```

Figura 1. Código para eliminar la columna índice.

- **Identificación y análisis de datos faltantes:**
  En un principio, se evidenció que había una proporción del 0,23% de datos faltantes, pero se logró descubrir que ese porcentaje no era real. Había un gran volumen de datos que estaban siendo tomados como ‘None’ en formato string, como una cadena de valores, lo cual hacía que tomara estos como datos reales y aportantes a la información. Para esto, se decide cambiar estos valores por None. La *Figura 2* muestra el código implementado para identificar la proporción de datos faltantes. La *Figura 3* muestra el código implementado para cambiar ‘None’ por None.

```
1 #Datos faltantes
2 faltantes = df.isnull().sum().sum()
3 print(f'La proporcion de datos faltantes es {round((faltantes/(len(df)*len(list(df.columns))))*100, 2)}%')

La proporcion de datos faltantes es 0.23%
```

Figura 2. Código para identificar datos faltantes.

```
[ ]  1 df = df.replace(to_replace='None', value=None)
```

Figura 3. Código para cambiar ‘None’ por None.

- **Coerción de variables:**
  Se realiza la transformación y coerción de las variables 'Date' y 'Did_Police_Officer_Attend_Scene_of_Accident', dado que estas se encontraban en formato 'object', complicando así su análisis. Dichas variables se transformaron en formato 'datetime' y 'int' respectivamente, ya que 'Date' hace referencia a la fecha que registra cada accidente y 'Did_Police_Officer_Attend_Scene_of_Accident' hace referencia a la presencia o la ausencia en la atención por parte de un oficial de la policía en cada accidente registrado. Además, se realiza la imputación de algunos datos faltantes, logrando así un nuevo valor de datos faltantes equivalente a 7,6%. La *Figura 4* muestra el código implementado para tratar las variables mencionadas. La *Figura 5* muestra el código para identificar la nueva proporción de datos faltantes.

```
[ ]  1 df['Date'] = pd.to_datetime(df['Date'], format = '%d/%m/%Y')
     2
     3 df['Did_Police_Officer_Attend_Scene_of_Accident'] = df['Did_Police_Officer_Attend_Scene_of_Accident'].replace(to_replace = 'Yes', value = 1)
     4 df['Did_Police_Officer_Attend_Scene_of_Accident'] = df['Did_Police_Officer_Attend_Scene_of_Accident'].replace(to_replace = 'No', value = 0)
     5 df['Did_Police_Officer_Attend_Scene_of_Accident'] = df['Did_Police_Officer_Attend_Scene_of_Accident'].astype(int)
```

Figura 4. Código implementado para tratar las variables.

```
1 print(f'La proporcion real de datos faltantes es {round((faltantes/(len(df)*len(list(df.columns))))*100, 2)}%')

La proporcion real de datos faltantes es 7.6%
```

Figura 5. Código para identificar nueva proporción de datos faltantes.

- **Imputación de datos que no son realmente faltantes:**
  Seguido, se aplicó una estrategia de imputación de datos faltantes en la variable 'Special_Conditions_at_Site' y 'Carriageway_Hazards'. Los datos vacíos fueron reemplazados por 'No special' y 'No hazards' respectivamente brindando así, una información más precisa de lo que desea compartir el dataset. Además, los datos vacíos de la variable 'Time' fueron remplazados por la hora 00:00 dejando así, una proporción de 1,48% de datos realmente faltantes. La *Figura 6* muestra el código para imputar dichos datos que no son considerados faltantes reales. La *Figura 7* muestra el código para identificar la proporción real de datos faltantes del dataset.

```
1 #En la var Special_condition_at_site Carriageway_Hazards los datos vacios
2 #puede ser reemplazados por 'No special' y 'No hazards' respectivamente
3 #Esto indicaria que no hay condicion especial en el sitio ni peligros.
4 df['Special_Conditions_at_Site'].fillna('No special', inplace = True)
5
6 df['Carriageway_Hazards'].fillna('No hazard', inplace = True)
7
8 #En la variable time, los datos faltantes corresponden a la hora 00:00
9 df['Time'].fillna('00:00', inplace = True)
```

Figura 6. Código para imputar datos faltantes.

```
1 print(f'La proporcion de datos faltantes tras estrategia de imputacion es {round((faltantes/(len(df)*len(list(df.columns))))*100, 2)}%')

La proporcion de datos faltantes tras estrategia de imputacion es 1.48%
```

Figura 7. Código para identificar la proporción real de datos faltantes.

- **Estandarización de variables:**
  Se decide estandarizar las variables de interés para comparar modelos con las variables estandarizadas y sin estandarizar buscando así, mejorar la interpretación de los datos. Se estandarizaron únicamente las siguientes variables:

  - *Accident_Severity*
  - *Number_of_Vehicles*
  - *Number_of_Casualties*
  - *Day_of_Week*
  - *Time*
  - *1st_Road_Number*
  - *Special_Conditions_at_Site*
  - *Carriageway_Hazards*
  - *Did_Police_Officer_Attend_Scene_of_Accident*

  Este proceso de estandarización logra mejorar la interpretación de los datos ya que se puede identificar patrones, en este caso categóricos. La *Figura 8* muestra el código para la estandarización de las variables.

```
#Se guardan las variables numericas en df2 y se estandarizan
df2 = df.copy()
df2 = df2.iloc[:,[6,7,8,10,11,17,28,29,31]] #Se incluye unicamente algunas vars cuantitativas de interes

# Forma de estandarización 1 usando stats.zscore de scipy
import pandas as pd
import scipy
from scipy import stats
for i in list(df2.columns):
#si i == Did_Police_Officer_Attend_Scene_of_Accident no estandariza ya que es 1 o 0
#si i == Time no estandariza ya que es la hora
  if i != list(df2.columns)[6] and i != list(df2.columns)[4]:
    df2[i] = stats.zscore(df2[i])
  else:
    pass

```

Figura 8. Estandarización de las variables de interés.

3. Modelos

3.1. Métodos supervisados

Se plantea un modelo con 4 variables regresoras las cuales son 'Speed_limit', 'Accident_Severity', 'Year', 'Did_Police_Officer_Attend_Scene_of_Accident' (Límite de velocidad, severidad del accidente, año, si el oficial atendió la escena del accidente), para predecir 'Number_of_Vehicles' (número de vehículos). Además, se dividen los datos X (variables predictoras) e Y (Variable respuesta) en datos de entrenamiento y testeo, utilizando el 20% de los datos para entrenar el modelo y randomizando la elección de los datos para entrenar. Para evaluar

el desempeño del modelo se escoge el MSE y MAPE, además también se calcula el R^2.

```
print(f'MSE en conjunto de entrenamiento: {mse_train}')
print(f'MSE en conjunto de prueba: {mse_test}')
print(f'R^2 en conjunto de entrenamiento: {r2_train}')
print(f'R^2 en conjunto de prueba: {r2_test}')
```

```
MSE en conjunto de entrenamiento: 0.5043004979122009
MSE en conjunto de prueba: 0.4992513956481341
R^2 en conjunto de entrenamiento: 0.014825922656813373
R^2 en conjunto de prueba: 0.014969275605214993
```

Figura 9. Impresión de desempeño1.

```
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

#y_test contiene los valores reales y y_test_pred contiene las predicciones
mape = mean_absolute_percentage_error(y_test, y_test_pred)
print(f'MAPE: {mape:.2f}%')
```

```
MAPE: 34.18%
```

Figura 10. Declaración función MAPE.

Además, se realiza el mismo procedimiento para los datos estandarizados, obteniendo los siguientes resultados:

```
print(f'MSE en conjunto de entrenamiento: {mse2_train}')
print(f'MSE en conjunto de prueba: {mse2_test}')
print(f'R^2 en conjunto de entrenamiento: {r2_train_2}')
print(f'R^2 en conjunto de prueba: {r2_test_2}')

#y2_test contiene los valores reales estadarizados y y2_test_pred contiene las predicciones
mape = mean_absolute_percentage_error(y2_test, y2_test_pred)
print(f'MAPE: {mape:.2f}%')
```

```
MSE en conjunto de entrenamiento: 0.9871221317954594
MSE en conjunto de prueba: 0.977238975599514 9
R^2 en conjunto de entrenamiento: 0.014825922656813262
R^2 en conjunto de prueba: 0.014969275605214438
MAPE: 99.39%
```

Figura 11.Impresión de desempeño2.

Se evidencia un desempeño bajo en el modelo con los datos estandarizados, por tanto, se decide tomar de referencia el modelo con los datos sin estandarizar (el cual tiene un MAPE de 34.18%) para tener un punto de comparación al momento de realizar el modelo no supervisado.

3.2. Métodos no supervisados

Para el modelo de métodos no supervisados, se elige realizar una red neuronal. Con el fin de comparar el resultado de dicha red, se decide darle como entrada al modelo las variables que fueron usadas como regresoras en los modelos de métodos supervisados, además dichos valores estarán sin estandarizar para

tener una comparación más directa del desempeño.

La red neuronal, consiste de 4 capas, una capa de entrada, con un input shape = 4 por las variables seleccionadas, 2 capas ocultas de 64 y 32 neuronas respectivamente, y una capa de salida donde se predecirá los valores del número de vehículos involucrados en el accidente. Para la activación de la capa de entrada y de las capas ocultas se utiliza 'Relu' dado la naturaleza de las variables y para la capa de salida se usa 'Linear'.

Respecto a los datos de entrenamiento nuevamente se separan en entrenamiento y testing de forma aleatoria, con el fin de usar diferentes registros que en los usados en los modelos anteriores. Al momento de entrenar la red, se define una tasa de ajuste del 10% y 10 épocas de entrenamiento, además de usar el MSE como función de pérdida y el MAPE como métrica.

```
model.compile(optimizer=tf.keras.optimizers.Adam(0.1),
    loss='mean_squared_error', metrics=['mape'])

# Entrenar el modelo
history = model.fit(X_train, y_train, epochs=10)
```

Figura 12. Compilación y ajuste de red neuronal.

Finalmente se obtiene un MAPE de 34.63% y un MSE de 0.5174, datos bastante cercanos al mejor modelo supervisado.

3.3. Curvas de aprendizaje

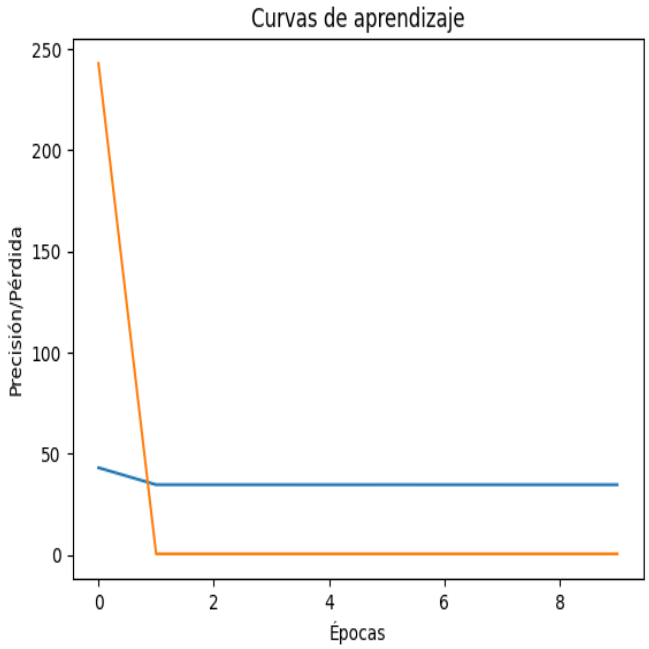

Figura 13. Curvas de aprendizaje.

En las curvas de aprendizaje se observa que la red neuronal encuentra el modelo definitivo de manera rápida y simplemente se queda oscilando entre modelos con pesos y sesgos muy similares con una función de pérdida estadísticamente igual, obteniendo un modelo en cuanto al desempeño similar al supervisado.

4. Retos y condiciones de despliegue del modelo

Los modelos de aprendizaje automático se pueden volver sumamente esenciales en la mejora de la seguridad vial, ofreciendo la capacidad de prever la cantidad de vehículos involucrados en accidentes. Este enfoque permite a las autoridades adoptar medidas preventivas para mitigar el riesgo de colisiones. No obstante, implementar un modelo predictivo para accidentes vehiculares plantea diversos desafíos y consideraciones cruciales.

En primer lugar, la precisión del modelo es fundamental. Validar el modelo con conjuntos de datos de prueba no utilizados en su entrenamiento es esencial para evitar el sobreajuste y garantizar una predicción precisa en situaciones reales. Dada la naturaleza impredecible de los accidentes, la validación con una amplia gama de datos de prueba se vuelve especialmente crucial.

En segundo lugar, es necesario asegurar que el modelo pueda escalar eficientemente para manejar el volumen previsto de datos. Dado que los accidentes son eventos relativamente raros, el modelo debe ser capaz de gestionar tanto conjuntos pequeños como grandes de datos de entrenamiento y prueba, garantizando su precisión en diversas condiciones.

El tercer desafío radica en mantener la relevancia del modelo a lo largo del tiempo. Dado que los accidentes pueden ser provocados por diversos factores cambiantes, como condiciones climáticas, diseño de carreteras y comportamiento humano, es esencial actualizar periódicamente el modelo para reflejar estos cambios y mantener su precisión.

Además, es imperativo asegurar que el hardware utilizado para ejecutar el modelo cumpla con los requisitos computacionales necesarios. Esto incluye aspectos como la cantidad de memoria, capacidad de procesamiento y almacenamiento. En el caso de modelos predictivos para accidentes vehiculares, podría ser necesario utilizar hardware de alto rendimiento, especialmente si el modelo es complejo o si se espera que maneje grandes volúmenes de datos. Asimismo, se debe garantizar que el software utilizado para ejecutar el modelo sea compatible tanto con el hardware como con el entorno de datos. Esto abarca aspectos como el sistema operativo, lenguaje de programación y bibliotecas de aprendizaje automático utilizadas. La compatibilidad con el formato de datos empleado para el entrenamiento y la prueba del modelo también es esencial.

5. Conclusiones

- Es importante ampliar el alcance de algunos modelos para que sean capaces de manejar variables cualitativas, cuantitativas y mixtas. Esto puede mejorar su rendimiento en tareas complejas.
- La evaluación del rendimiento del modelo es esencial para garantizar su precisión y fiabilidad. Se deben utilizar métricas adecuadas y realizar pruebas en conjuntos de datos de prueba o validación cruzada para evaluar la capacidad del modelo para generalizar a nuevos datos.
- El análisis de la importancia de las variables es fundamental para mejorar el rendimiento de los modelos. Las variables con mayor peso pueden ser identificadas para su optimización, lo que puede conducir a una reducción del error.
- La gestión de datos faltantes o inconsistentes es fundamental para evitar sesgos en el modelo. Estrategias como la imputación o eliminación resultan sumamente útiles.
- El modelo que seleccionó la red neuronal que se implementó, arroja unas métricas (MSE y MAPE) muy cercanas o similares a dichas métricas del mejor modelo supervisado implementado.

Bibliografía

- *Road Accident (United Kingdom (UK)) dataset*. (2022, 28 mayo). Kaggle. https://www.kaggle.com/datasets/devansodariya/road-accident-united-kingdom-uk-dataset/data